# Systèmes de recommandation

Raphaël Fournier-S'niehotta

Sorbonne Université, raphael.fournier@lip6.fr

IP Paris
CES-IA
2024-2025

SORBONNE UNIVERSITÉ

# Plan

# Introduction

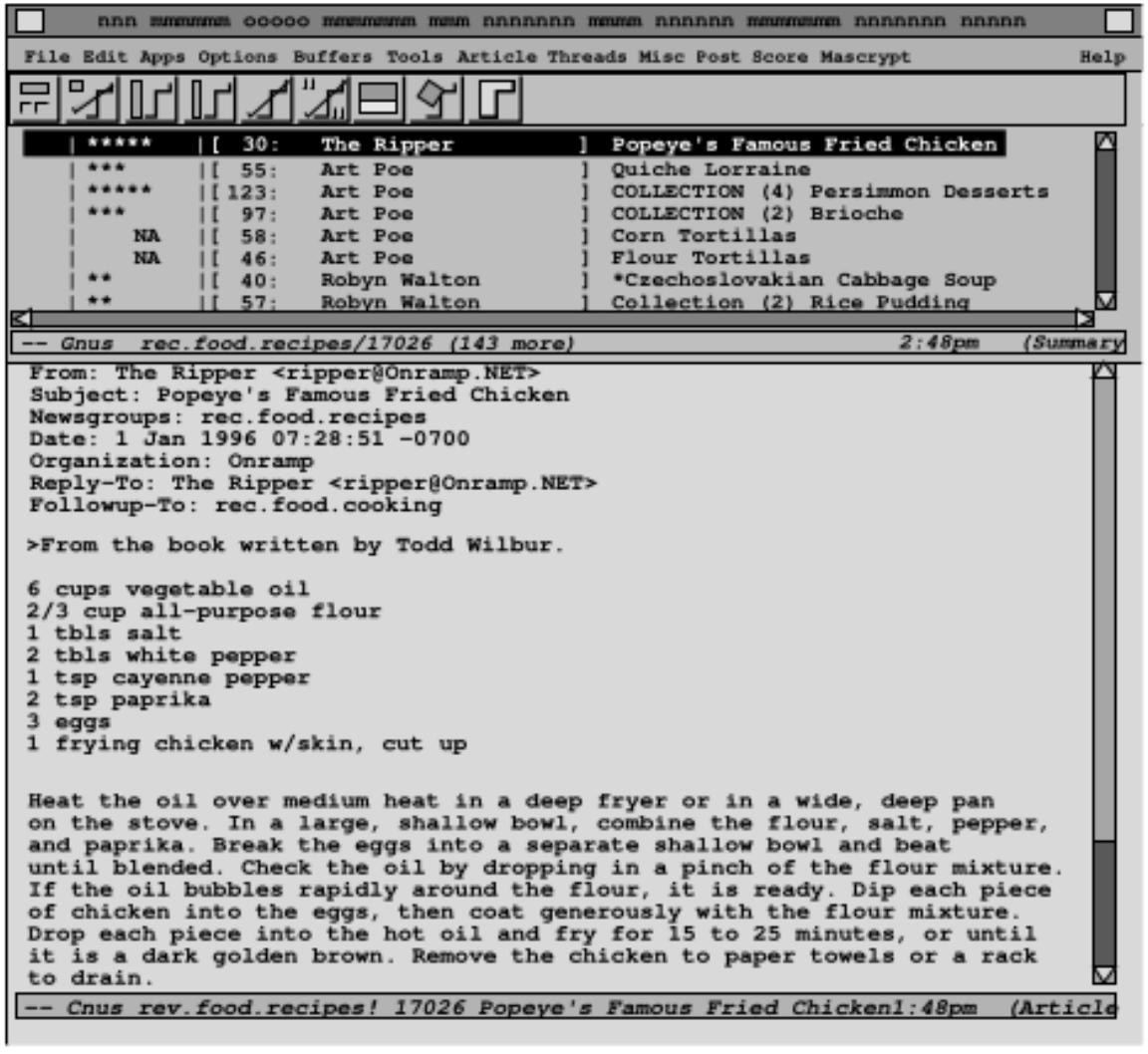
# Plan

# Définition

Un *système de recommandation* **filtre** les éléments (*items*) d'une collection à des utilisateurs (*users*).

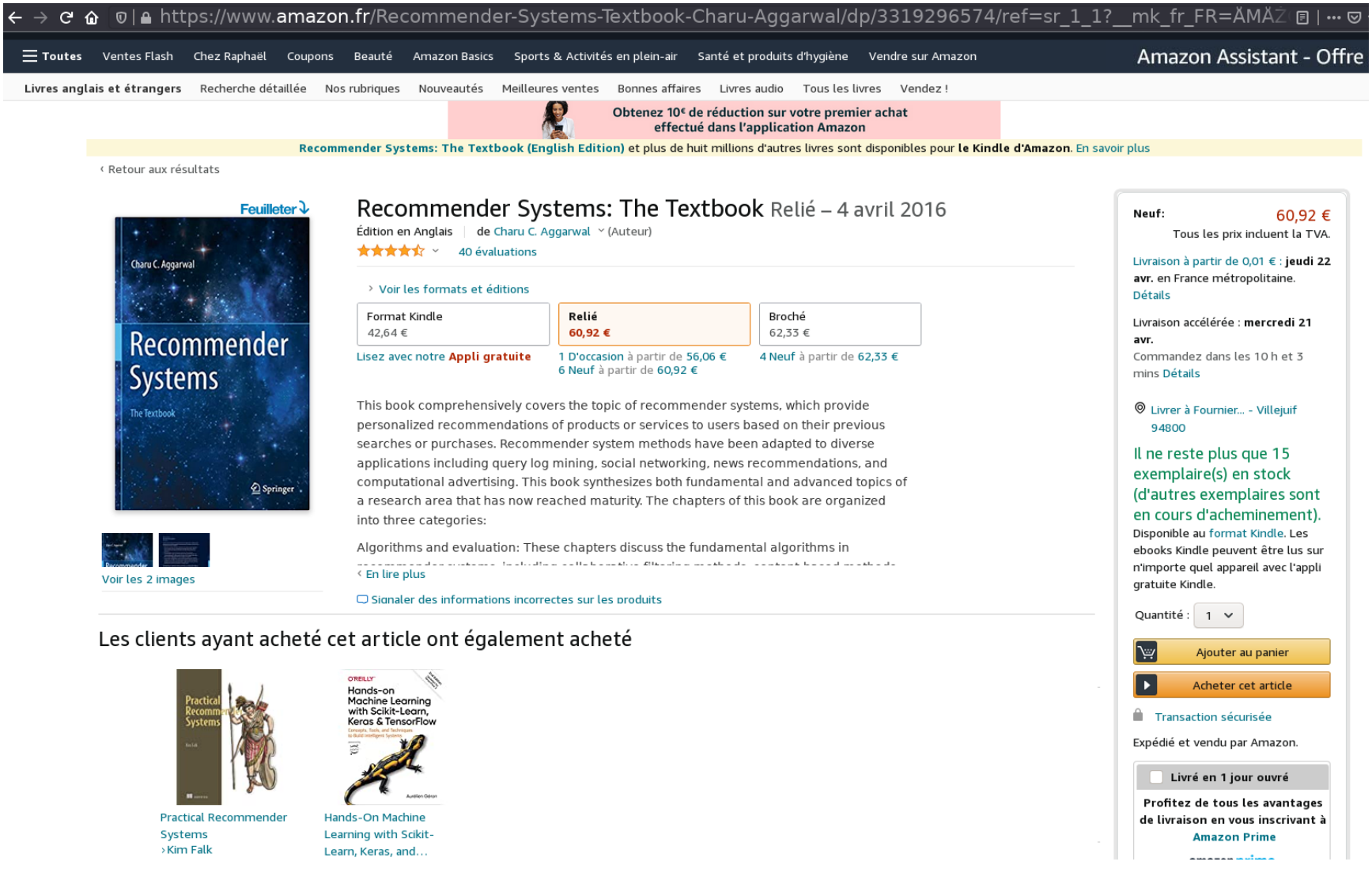
- Utilisateurs·ices : presque toujours des personnes
- Éléments :
  - commerciaux (produits à vendre)
  - culturels (films, chansons, presse)
  - professionnels (articles scientifiques)

# Quelques exemples



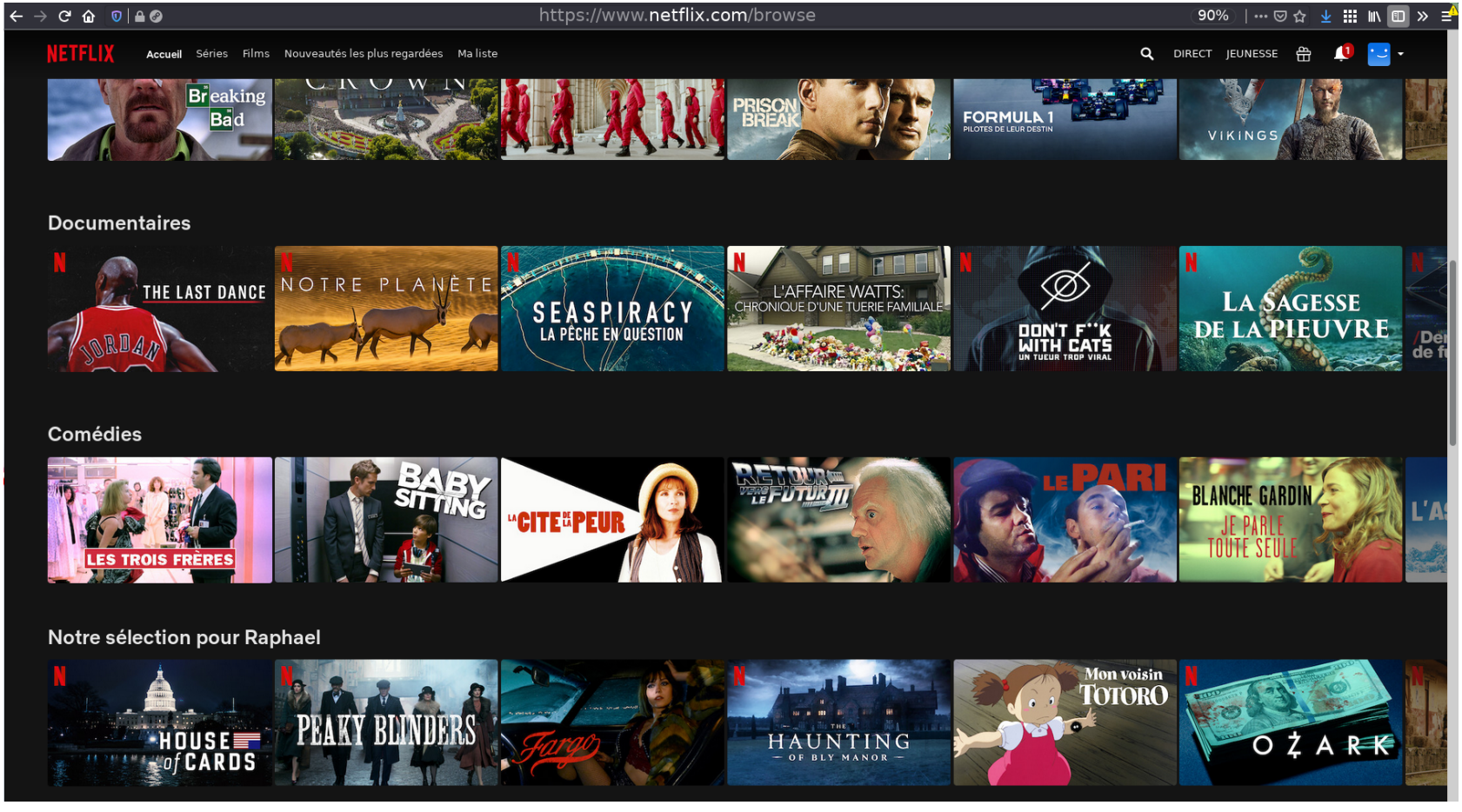

- Usenet news
- Groupe de recherche GroupLens, à l'origine de datasets importants dans la communauté et de nombreux algorithmes.

# Quelques exemples : Amazon



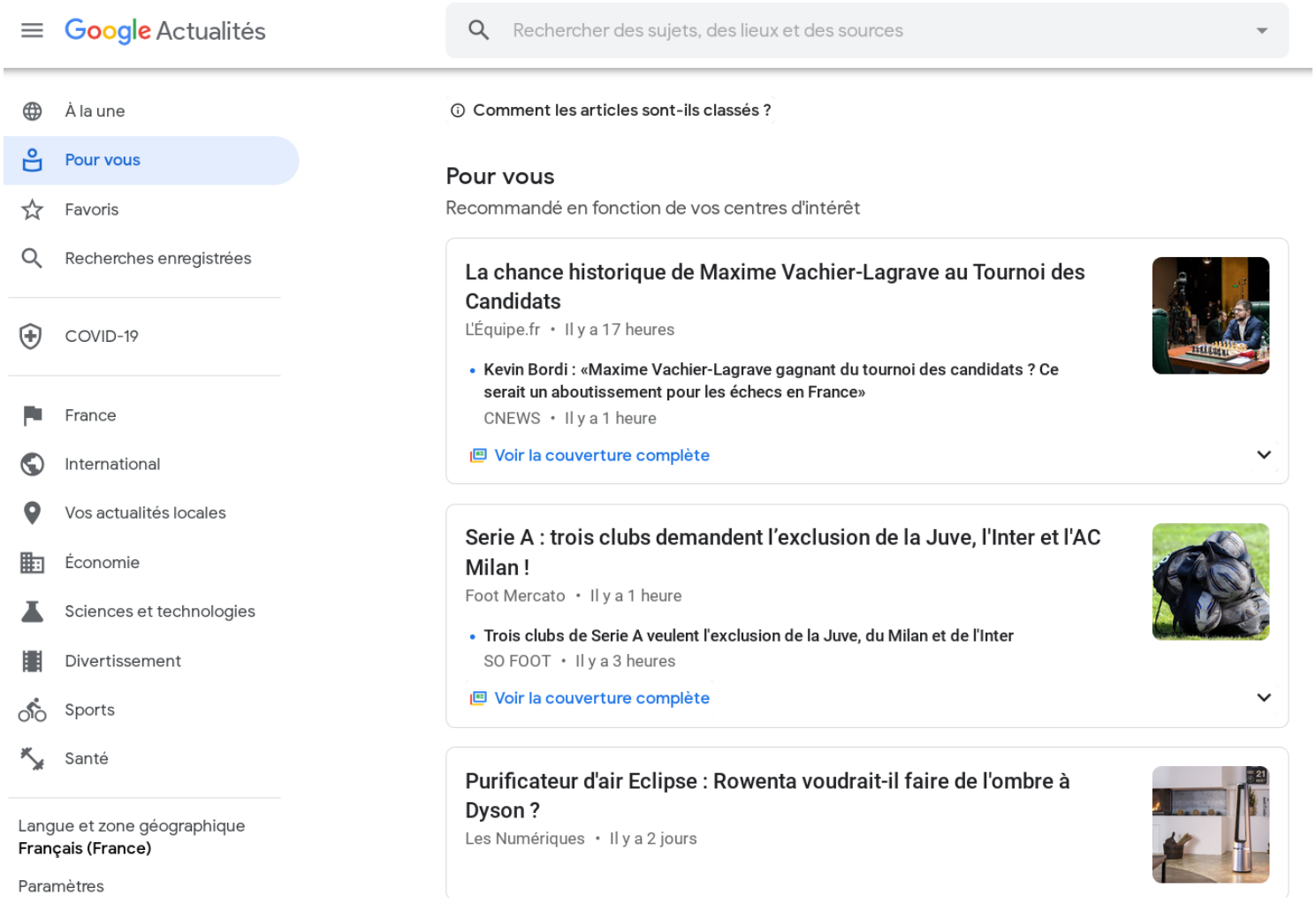

- Pionniers dans le domaine commercial
- échelle 1-5, explications, implicit/explicit

# Quelques exemples : Netflix

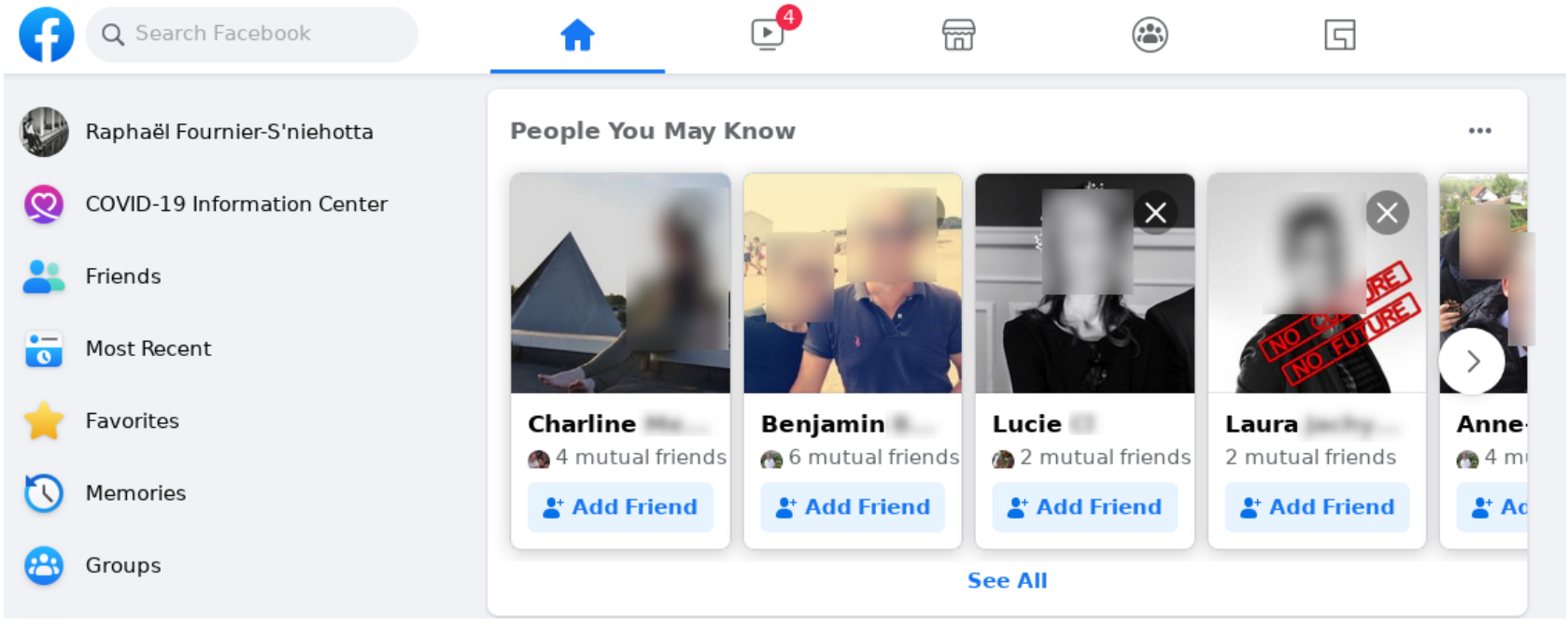



- Explications détaillées, Netflix Prize

# Quelques exemples : Google Actualités (news)



# Quelques exemples : Facebook




- Vise l'accroissement de la taille du réseau (pas directement des "ventes")

# Mais aussi

| | |
|---|---|
| Spotify, Deezer | Musique |
| LinkedIn | Réseaux Pro |
| Yelp | Restaurants |
| TripAdvisor | Voyages |
| Youtube | Vidéos en ligne |

# Motivation

- ”Ces systèmes guident les gens vers des contenus intéressants grâce aux recommandations d’autres personnes.” (Paul Resnick)

Quels problèmes cherche-t-on à résoudre ?

- surcharge d’information
- longue traîne
- expérience utilisateur
- reproduction d’un modèle hors ligne

# Objectifs, défis

- Objectif premier : accroissement des ventes (contexte commercial)
- Manières plus ou moins directes !
  - Facilité d'accès aux contenus du système
  - Améliorer la pertinence de ce que l'utilisateur voit
  - Nouveauté
  - Surprise (légèrement différent)
  - Diversité

Défis :

- Taille des matrices
- Matrices creuses
- Diversité des objectifs à satisfaire

# Interdisciplinarité

Le problème de la recommandation est proche de plusieurs disciplines :

- psychologie (expression du besoin, des goûts)
- marketing (accroissement des ventes)
- design (interfaces, graphiques ou non)
- statistiques (probabilités, modélisation)
- informatique (passage à l'échelle)

# Modélisation du problème

- De très nombreuses approches existent !
- Les deux plus importantes : ”prédiction” et ”classement”

## Prédiction

- On cherche à prédire l'évaluation qu'un utilisateur donnerait à un élément.
- Avec $n$ items et $m$ users, on a une matrice $m \times n$
- les données d'apprentissage sont les cases remplies,
- on cherche à remplir les cases vides

## Classement

- On ne cherche plus l'évaluation d'un élément par une personne
- On détermine les $k$ éléments les plus intéressants (**top-$k$**)
- Seul l'ordre compte (pas les écarts entre éléments)
- On peut utiliser une modélisation ”prédiction” puis trier, pour faire un classement

# Modélisation générique

| | i1 | i2 | i3 | i4 |
|---|---|---|---|---|
| u1 | | 3 | 5 | 2 |
| u2 | 1 | 5 | 2 | 4 |
| u3 | | 3 | | |
| u4 | | | 5 | 2 |
| u5 | | 1 | 4 | |

| | i1 | i2 | i3 | i4 |
|---|---|---|---|---|
| u1 | **1** | 3 | 5 | 2 |
| u2 | 1 | 5 | 2 | 4 |
| u3 | **2** | 3 | **1** | **4** |
| u4 | **1** | **4** | 5 | 2 |
| u5 | **2** | 1 | 4 | **3** |

| | |
|---|---|
| u1 | i1 1 |
| u3 | i4 4 |
| | i1 2 |
| | i3 1 |

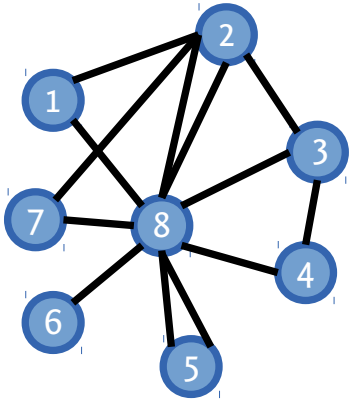
---

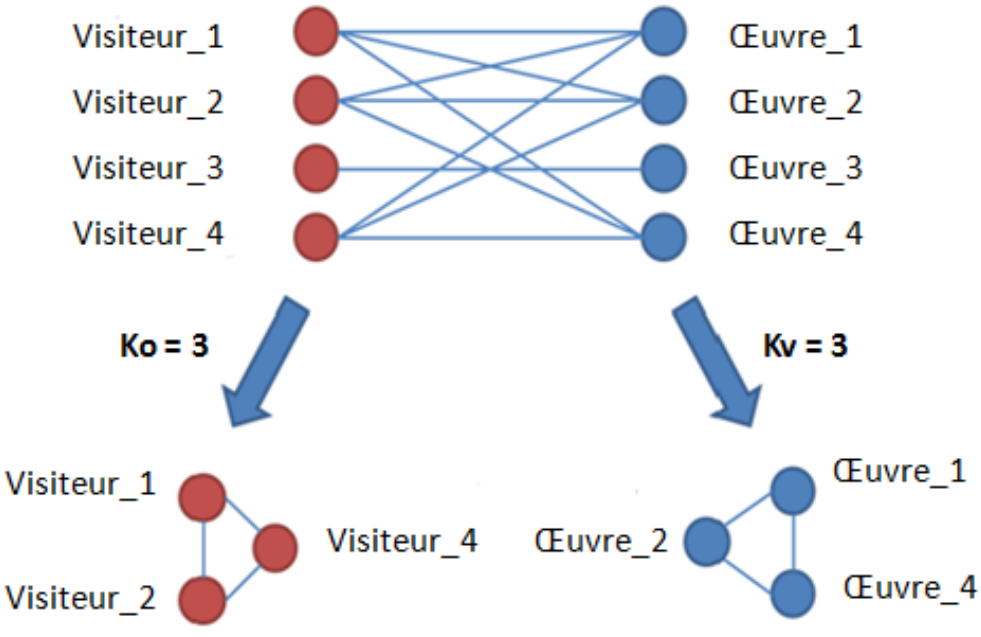
| | i1 | i2 | i3 | i4 |
|---|---|---|---|---|
| u1 | | 1 | 1 | 1 |
| u2 | 1 | 1 | 1 | 1 |
| u3 | | 1 | | |
| u4 | | | 1 | 1 |
| u5 | | 1 | 1 | |

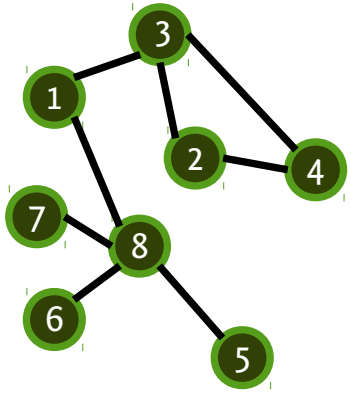
| | i1 | i2 | i3 | i4 |
|---|---|---|---|---|
| u1 | **0** | 1 | 1 | 1 |
| u2 | 1 | 1 | 1 | 1 |
| u3 | **0** | 1 | **1** | **1** |
| u4 | **0** | **0** | 1 | 1 |
| u5 | **1** | 1 | 1 | **0** |

| | |
|---|---|
| u3 | i3 |
| | i4 |
| u5 | i1 |

# Modélisation générique




graphes de données *implicites* ou *explicites*

# Données : échelles de valeurs

- Le système de recommandation que l'on *peut* mettre en place dépend beaucoup des données dont on dispose.
- Les évaluations (*ratings*) sont souvent sur une échelle de valeurs
- Parfois continue, le plus souvent discrète
- Valeurs négatives parfois autorisées ($\{-2,-1,0,1,2\}$, $[-10;10]$)
- Systèmes les plus courants : échelle de 1 à 5, ou 1 à 10
- Interprétation variable, échelles équilibrées (o/n), élement neutre (o/n)

- Valeurs catégorielles ordonnées
- Binarité 0-1
- Unarité

# Plan

# Données : réactions explicites et implicites

| | GLADITOR | GODFTHER | BEN-HUR | GOODFELLAS | SCARFACE | SPARTACUS |
|---|---|---|---|---|---|---|
| $U_1$ | 1 | | | 5 | | 2 |
| $U_2$ | | 5 | | | 4 | |
| $U_3$ | 5 | 3 | | 1 | | |
| $U_4$ | | | 3 | | | 4 |
| $U_5$ | | | | 3 | 5 | |
| $U_6$ | 5 | | 4 | | | |

(a) Ordered ratings

| | GLADIATOR | GODFATHER | BEN-HUR | GOODFELLAS | SCARFACE | SPARTACUS |
|---|---|---|---|---|---|---|
| $U_1$ | 1 | | | 1 | | 1 |
| $U_2$ | | 1 | | | 1 | |
| $U_3$ | 1 | 1 | | 1 | | |
| $U_4$ | | | 1 | | | 1 |
| $U_5$ | | | | 1 | 1 | |
| $U_6$ | 1 | | 1 | | | |

(b) Unary ratings

- L'expressivité de la 2e est plus faible : on ne sait pas que U1 n'aime pas Gladiator.
- La 2e permet seulement d'exprimer des préférences "positives"

# Données et sur-apprentissage

- Données unaires, implicit feeback
  - Analyse simplifiée en identifiant 0 et données manquantes
  - Sans ça, sur-apprentissage (pas assez de valeurs distinctes)
  - Biais introduit, mais assez faible
    (contexte : la plupart des choses ne seront pas vues)
  - En général très faciles à collecter !
  - Nombreuses sources (achat, visite de page, *actions*)
- Données explicites, échelles plus grandes
  - analyse non simplifiée en identifiant 0, biais plus fort
  - Données plus délicates à collecter : contribution volontaire de l'utilisateur
  - Incitations explicites (questions, récompenses, etc.)
  - Incitations implicites (design d'interface)

# Données : critiques

- Généralisation des systèmes de recommandation ces dernières années
- La dimension humaine est parfois oubliée
- Attention à ce qu'on note : Uberisation

- Crédit social, assurances : croisement de sources
- Sécurité, évidemment
- droit à l'oubli

- Weapons of Math Destruction (Cathy O'Neil, 2016)
- BlackMirror
- David Chavalarias, Toxic Data
- Éric Sadin, La vie algorithmique

# La suite : les grands modèles de RS

Historiquement, 4 grandes familles de systèmes de recommandation :

1. le filtrage collaboratif, dont :
   - les méthodes ”à mémoire” ou ”avec voisinage” (*neighbour-based collaborative filtering*)
   - les méthodes à modèle (*model-based collaborative filtering*)
2. le filtrage par le contenu (*content-based methods*)
3. les méthodes reposant sur des connaissances externes (*knowledge-based methods*)
4. les systèmes hybrides, combinant des approches précédentes (*ensemble/hybrid methods*)

- Apports récents : les LLM (cf plus tard)

# Filtrage collaboratif par le voisinage

# Filtrage collaboratif

- Par le voisinage (*neighbour-based collaborative filtering*)
  - User-based
  - Item-based

## User-based

- On cherche des utilisateurs *similaires* à A
- on recommande des éléments à A en faisant une moyenne pondérée des notes de ces utilisateurs.
- On se limite en général à $k$ utilisateurs.
- Les fonctions de similarité sont à calculer sur les lignes de la matrice des notes.

## Item-based

- Pour connaître la note de A sur l'item I, on cherche un ensemble d'items similaires à I, que A a noté, et on calcule une moyenne pondérée.
- Les similarités sont calculées sur les colonnes.

# Filtrage collaboratif par le voisinage

## Principe élémentaire

- il est possible de trouver des relations (dépendances, corrélations) dans les interactions utilisateurs–items
- Un utilisateur intéressé par un documentaire historique a plus de chance d'être intéressé par un autre documentaire historique ou des programmes éducatifs qu'un film d'action
- Des catégories d'éléments présentent des corrélations significatives, que l'on peut utiliser pour faire des recommandations plus précises
- Cela peut aussi se jouer à une granularité plus fine que les catégories

- Présuppose que l'on peut apprendre ces relations, grâce aux données
- Plus on a de données, plus les recommandations seront robustes
- Les modèles peuvent identifier des catégories d'utilisateurs aux comportements de consommation similaires

# Avantages, inconvénients

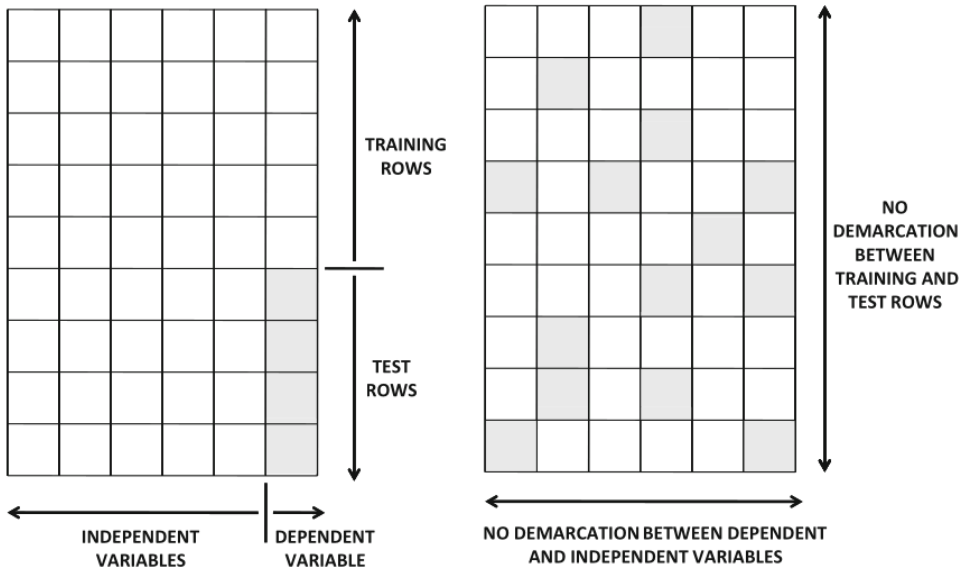
## Avantages

- très simple à implémenter
- recommandations faciles à expliquer

## Inconvénients

- peu efficace avec matrice très creuses
- trop peu d'utilisateurs ou d'items similaires : *couverture* faible
- souvent, seuls les top-$k$ éléments suffisent, ce n'est pas toujours un gros problème

# Proximité avec la classification



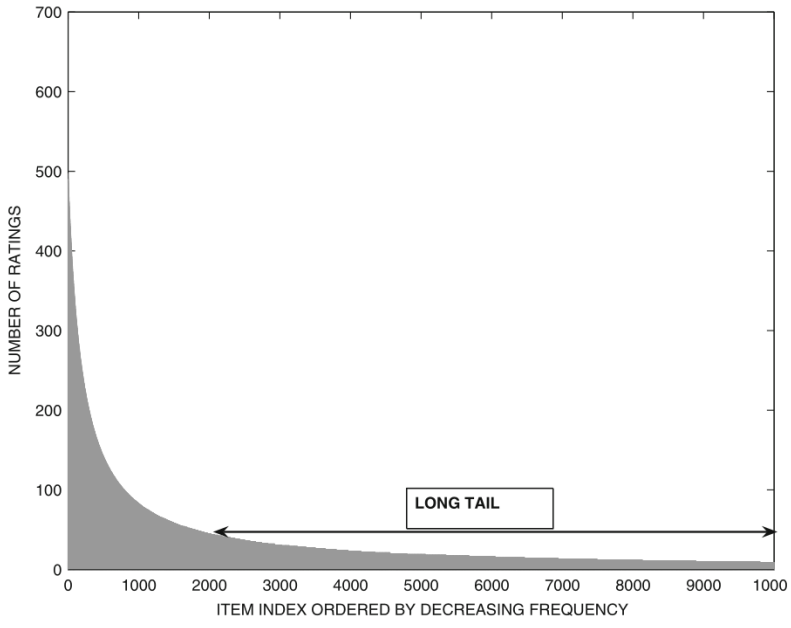

(a) Classification (b) Collaborative filtering

- Analyse de données manquantes
  - cas difficile, large, sparse

# Inductif, transductif

- Dans un contexte *transductif*, les données de test sont incluses dans les données d'entraînement
- Il est souvent dur de faire des prédictions pour des données de test non présentes lors de l'entraînement
- Les modèles capables de faire des prédictions pour de nouvelles instances sont dits *inductifs* (naïve Bayes)
- L'enchevêtrement des données dans la matrice en recommandation fait qu'on est en transductif

# Longue traîne et matrices




- Intérêt commercial limité pour les items fréquents
  (puisqu'ils sont déjà très vendus)
- Recommander la long tail
- Difficulté d'avoir des notes, suggestion d'items populaires
- Baisse de diversité
- Difficulté de calculer des voisinages fiables

# Plan

# User-based CF

Principe : les utilisateurs similaires ont des évaluations similaires pour des éléments donnés.

- On cherche un *voisinage* d'utilisateurs similaires à l'utilisateur cible
- On calcule la similarité entre cet utilisateur et tous les autres
- on en garde quelques uns

Difficile car :

- différentes manières de noter : certains notent ce qu'ils aiment (surtout), d'autres ce qu'ils n'aiment pas
- il faut que les ensembles notés coïncident

# Notation pour la matrice

Soit $R = r_{uj}$ la matrice $m \times n$.
$I_u$ est l'ensemble des indices des items notés par $u$.

L'ensemble noté par $u$ et $v$ est : $I_u \cap I_v$.

Exemple :
$I_u = \{1,3,5\}$, $I_v = \{1,2,3,4\}$, $I_u \cap I_v = \{1,3\}$.

# Similarité : coefficient de corrélation de Pearson (PCC)

- On calcule d'abord la moyenne des notes de $u$, $\mu_u$ :

$$\mu_u = \frac{\sum_{k \in I_u}}{|I_u|} \qquad \forall u \in \{1...m\}$$

- Puis la similarité elle-même :

$$\text{Sim(u,v)} = \frac{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} (r_{vk} - \mu_v)^2}} \qquad \forall (u, v) \in \{1...m\}^2$$

- On pourrait garder le top-$k$ des utilisateurs avec le meilleur PCC
- Mais nombre très variable de ratings pour un élément donné
- On cherche les top-$k$ pour chaque élément

# Étape 2 : calcul de la note

- Manière simple : on fait une moyenne pondérée des notes des utilisateurs
- Éventuellement les $k$ plus proches
- Poids : les PCC entre utilisateurs

- Problème : les échelles de notes individuelles
- Solution : évaluations centrées

$$s_{uj} = r_{uj} - \mu_u \qquad \forall u \in \{1...m\}$$

- On calcule une évaluation pondérée sur ces notes (positive ou négative)
- La note finale prédite $\hat{r}_{uj}$ est obtenue en rajoutant la moyenne :

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P(j)} \text{Sim(u,v)} \cdot s_{uj}}{\sum_{v \in P(j)} |\text{Sim(u,v)}|}$$

# Exemple UBCF

On cherche $\hat{r}_{31}$ et $\hat{r}_{36}$ dans l'exemple suivant (notes $[0;7]$) :

| Item-Id ⇒ / User-Id ⇓ | 1 | 2 | 3 | 4 | 5 | 6 | Mean Rating | Cosine$(i,3)$ (user-user) | Pearson$(i,3)$ (user-user) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 6 | 7 | 4 | 5 | 4 | 5.5 | 0.956 | 0.894 |
| 2 | 6 | 7 | ? | 4 | 3 | 4 | 4.8 | 0.981 | 0.939 |
| 3 | ? | 3 | 3 | 1 | 1 | ? | 2 | 1.0 | 1.0 |
| 4 | 1 | 2 | 2 | 3 | 3 | 4 | 2.5 | 0.789 | -1.0 |
| 5 | 1 | ? | 1 | 2 | 3 | 3 | 2 | 0.645 | -0.817 |

$$\text{Cosine}(1,3) = \frac{6*3+7*3+4*1+5*1}{\sqrt{6^2+7^2+4^2+5^2}\cdot\sqrt{3^2+3^2+1^2+1^2}} = 0.956$$

$$\text{Pearson}(1,3) =$$

$$= \frac{(6-5.5)*(3-2)+(7-5.5)*(3-2)+(4-5.5)*(1-2)+(5-5.5)*(1-2)}{\sqrt{1.5^2+1.5^2+(-1.5)^2+(-0.5)^2}\cdot\sqrt{1^2+1^2+(-1)^2+(-1)^2}}$$

$$= 0.894$$

- PCC plus discriminant que Cosinus
- Le signe peut être informatif : (dis)similarité
- top-2 utilisateurs proches de $u_3$ : $u_1$ et $u_2$

# Exemple UBCF

- Sans centrer :

$$\hat{r}_{31} = \frac{7 * 0.894 + 6 * 0.939}{0.894 + 0.939} \approx 6.49$$

$$\hat{r}_{36} = \frac{4 * 0.894 + 4 * 0.939}{0.894 + 0.939} = 4$$

- On privilégierait donc l'item 1, plutôt que le 6
- les prédictions suggèrent que $u_3$ aimera 1 et 6 plus que *tous* les autres.
- attention : biais, $\{u_1, u_2\}$ est un groupe optimiste.

- En centrant :

$$\hat{r}_{31} = 2 + \frac{1.5 * 0.894 + 1.2 * 0.939}{0.894 + 0.939} \approx 3.35$$

$$\hat{r}_{36} = 2 + \frac{-1.5 * 0.894 - 0.8 * 0.939}{0.894 + 0.939} \approx 0.86$$

- même résultat, 1 mieux que 6
- mais ! $r_36 = 0.86$, moins que les autres ratings
- $u_1$ et $u_2$ avaient aussi noté 6 moins que les autres

Attention : prédictions pour 6 éventuellement hors de l'intervalle des valeurs

# Amplification des poids

- on amplifie parfois le poids des utilisateurs dans la moyenne pondérée

$$\text{Sim(u,v)} = \text{Pearson(u,v)}^{\alpha} \qquad \text{avec } \alpha > 1$$

- On peut aussi filtrer le ”voisinage” (*peer group*) d’un utilisateur de différentes façons
- on peut éliminer des notes faiblement ou négativement corrélées

# Impact de la longue traîne

- Les éléments populaires apparaissent souvent dans les ensembles d'éléments en commun entre deux utilisateurs
- Pourtant, ils sont peu discriminants
- Recommandation de moindre qualité (sélection du voisinage, calcul de note)

Solution (cf *Information retrieval*) :

- Termes peu informatifs et fréquents (*stop-words*, "le", "un·e", "de", etc.) évacués avec Inverse Document Frequency (tf.idf)
- Ici : Inverse User Frequency. Pour chaque élément $j$, noté $m_j$ fois, on définit :

$$w_j = log(\frac{m}{m_j}) \qquad \forall j \in \{1...n\}$$

- Ce poids est introduit dans le PCC :

$$\text{Pearson(u,v)} = \frac{\sum_{k \in I_u \cap I_v} w_k (r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} w_k \cdot (r_{uk} - \mu_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} w_k \cdot (r_{vk} - \mu_v)^2}}$$

# Plan

# Item-based CF

- Les voisinages sont construits en termes d'items, et non d'utilisateurs
- Comme précédemment, on calcule des évaluations centrées $s_{uj}$
- On définit l'ensemble $U_i$ des utilisateurs qui ont évalué $i$
- On calcule une similarité cosinus :

$$cosA(i,j) = \frac{\sum_{u \in U_i \cap U_j} s_{ui} \cdot s_{uj}}{\sqrt{\sum_{u \in U_i \cap U_j} s_{ui}^2} \cdot \sqrt{\sum_{u \in U_i \cap U_j} s_{uj}^2}}$$

- Un PCC entre colonnes pourrait aussi être utilisé, mais les résultats sont généralement moins bons

# Item-based CF

- Pour prédire la note $\hat{r}_{ut}$ de l'utilisateur $u$ sur l'objet $t$, on calcule :
  - un ensemble $Q_t(u)$ contenant les top-$k$ éléments
  - puis :

$$\hat{r}_{ut} = \frac{\sum_{j \in Q_t(u)} cosA(j,t) \cdot r_{uj}}{\sum_{j \in Q_t(u)} |cosA(j,t)|}$$

- On utilise les propres notes de l'utilisateur pour faire la prédiction

- Dans le cas de films, les top-$k$ éléments seront souvent des films du même genre

# Item-based CF

On cherche $\hat{r}_{31}$ et $\hat{r}_{36}$ :

| Item-Id ⇒ / User-Id ⇓ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1.5 | 0.5 | 1.5 | -1.5 | -0.5 | -1.5 |
| 2 | 1.2 | 2.2 | ? | -0.8 | -1.8 | -0.8 |
| 3 | ? | 1 | 1 | -1 | -1 | ? |
| 4 | -1.5 | -0.5 | -0.5 | 0.5 | 0.5 | 1.5 |
| 5 | -1 | ? | -1 | 0 | 1 | 1 |
| Cosine$(1, j)$ (item-item) | 1 | 0.735 | 0.912 | -0.848 | -0.813 | -0.990 |
| Cosine$(6, j)$ (item-item) | -0.990 | -0.622 | -0.912 | 0.829 | 0.730 | 1 |

$$\text{AdjustedCosine}(1,3) = \frac{1.5 * 1.5 + (-1.5) * (-0.5) + (-1) * (-1)}{\sqrt{1.5^2 + (-1.5)^2 + (-1)^2} \cdot \sqrt{1.5^2 + (-0.5)^2 + (-1)^2}} = 0.912$$

- les items 2 et 3 sont les plus proches de 1
- les items 4 et 5 sont les plus proches de 6

# Item-based CF

On obtient :

$$\hat{r}_{31} = \frac{3 * 0.735 + 3 * 0.912}{0.735 + 0.912} = 3$$
$$\hat{r}_{36} = \frac{1 * 0.829 + 1 * 0.730}{0.829 + 0.730} = 1$$

- Évaluations dans l'intervalle et plus en phase avec les autres notes de l'utilisateur
- En général les listes de recommandations UBCF et IBCF sont proches

# Plan

# Complexité

- Approche gloutonne : on calcule les notes de tous les éléments pour un utilisateur donné, puis on classe.
- Possible, mais coûteux
- Réutilisation de calculs intermédiaires

En pratique : une phase *offline* et une phase *online* :

- En offline, les similarités entre utilisateurs et voisinage sont pré-calculés
- En online, on calcule les notes et on classe

# Complexité

- Soit $n' \ll n$ le nombre max d'évaluations d'items par les utilisateurs
- Soit $m' \ll m$ le nombre max d'évaluations reçues par un élément
- Calculer d'un voisinage en $\mathcal{O}(m \cdot n')$, de tous en $\mathcal{O}(m^2 \cdot n')$
- En item-based : $\mathcal{O}(n^2 \cdot m')$
- En online, chaque note est calculée en $\mathcal{O}(k)$, $k$ taille du voisinage considéré
- Donc $\mathcal{O}(k \cdot n)$ pour toutes, avant classement (tri).

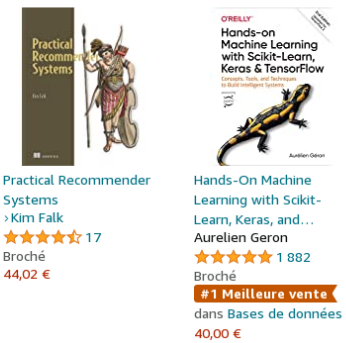
# Plan

# Quelques éléments de comparaison

- Utiliser *les propres notes* de l'utilisateur est généralement meilleur que *les notes des autres utilisateurs similaires*
- Mais cela peut conduire à privilégier des éléments trop (ou toujours) similaires à ceux déjà vus : moins de diversité et de surprise !
- Stabilité meilleure en IB. Plus d'users que d'items, donc les items auront des ensembles d'users en commun plus larges. Et plus d'utilisateurs arrivent dans le système, donc moins de recalcul de voisinage.
- Explicabilité meilleure en item-based :

*Because you watched "Secrets of the Wings," [the recommendations are]* ⟨*List*⟩ .

Les clients ayant acheté cet article ont également acheté




- En user-based, on ne connaît pas les *utilisateurs similaires*

# Avantages et inconvénients de ces méthodes

## Avantages

- Facilité d'implémentation, de débuggage
- Interprétabilité raisonnable (vs méthodes à modèle)
- Stabilité

## Inconvénients

- Phase offline couteuse en temps et espace
- Couverture limitée (matrice creuse)

# Réduction de dimension

- On cherche une représentation à base de facteurs latents dans un espace de basse dimension *latent factor models*
- une distance entre facteurs latents d'utilisateurs peut-être calculée, même avec peu d'items communs
- efficacité accrue pour les calculs de voisinages

Deux possibilités :

- réduction de dimension sur les lignes **ou** les colonnes de la matrice
- réduction sur les deux simultanément (cf plus tard)
- Méthodes possibles : SVD ou ACP

# Réduction de dimension : SVD

- Augmentation de valeurs manquantes de la matrice (avec les moyennes de ligne ou colonne) : on obtient une matrice $R_f$
- On calcule une matrice de similarité $S = R_f^T R_f$
- S est semi-définie positive, peut s'écrire :

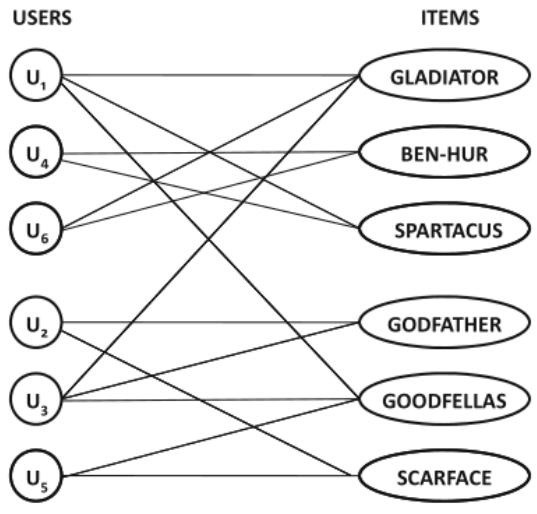
$$S = P \Delta P^T$$

- Avec :
  - P est une matrice $n \times n$, avec les vecteurs propres de S (orthonormés)
  - $\Delta$ est une matrice diagonale avec les valeurs propres non nulles de S
- Soit $P_d$ une matrice $n \times d$ contenant les vecteurs correspondant aux valeurs propres les plus grandes
- La représentation réduite de $R_f$ est $R_f P_d$ $(m \times d)$
- Chaque utilisateur a une représentation en dimension $d$
- On l'utilise pour calculer les voisinages, avant calcul de prédictions.

# Plan

# Graphes

| | GLADIATOR | GODFATHER | BEN-HUR | GOODFELLAS | SCARFACE | SPARTACUS |
|---|---|---|---|---|---|---|
| $U_1$ | 1 | | | 5 | | 2 |
| $U_2$ | | 5 | | | 4 | |
| $U_3$ | 5 | 3 | | 1 | | |
| $U_4$ | | | 3 | | | 4 |
| $U_5$ | | | | 3 | 5 | |
| $U_6$ | 5 | | 4 | | | |




- Les graphes disposent d'outils et de concepts dédiés, que l'on peut mobiliser
- Variétés d'algorithmes différents
- Problème lié à la notion de *prédiction de liens* en SNA

# Marches aléatoires, chemins

## Marches aléatoires

- Méthodes probabilistes, telles que le Personal PageRank ou SimRank
- Efficacité car connexité peut-être large dans un graphe

## Chemins

- Affinité basée sur les nombres de chemins et leurs longueurs :
- Centralité de Katz

$$\text{Katz}(i,j) = \sum_{t=1}^{\infty} \beta^t n_{ij}^{(t)}$$

  - $n_{ij}^{(t)}$ nombre de chemins entre $i$ et $j$
  - $\beta < 1$ paramètre
    (inférieur à l'inverse de la plus grande v.p de A)

- La matrice des centralités Katz $K$ se calcule :

$$K = \sum_{i=1}^{\infty} (\beta A)^i = (I - \beta A)^{-1} - I$$

# Filtrage collaboratif par modèle

# Plan

# Filtrage collaboratif par modèle

- Dans le FC par voisinage, pas de modèle, les données orientent l'apprentissage (*instance-based learning*)

- Ici :
  1. apprentissage (*training*), une construction du modèle
     - Apprentissage *classique* : arbre de décision, règles, classifieur Bayes, réseaux de neurones
  2. phase distincte de prédiction

# Règles d'association

- Historiquement, règles d'association proposées dans le contexte du supermarché
- On considère un ensemble de transactions $T = \{T_1 ... T_m\}$ définies sur $n$ éléments de $I$.
- Chaque T est un sous-ensemble d'éléments de I, on cherche les corrélations de sous-ensembles

- Exemple : {Bread, Butter, Milk} et {Fish, Beef, Ham} sont fréquents
- Mary a acheté {Butter, Milk}, on peut penser qu'elle achètera Bread

| Item ⇒ / Customer ⇓ | *Bread* | *Butter* | *Milk* | *Fish* | *Beef* | *Ham* |
|---|---|---|---|---|---|---|
| Jack | 1 | 1 | 1 | 0 | 0 | 0 |
| Mary | 0 | 1 | 1 | 0 | 1 | 0 |
| Jane | 1 | 1 | 0 | 0 | 0 | 0 |
| Sayani | 1 | 1 | 1 | 1 | 1 | 1 |
| John | 0 | 0 | 0 | 1 | 0 | 1 |
| Tom | 0 | 0 | 0 | 1 | 1 | 1 |
| Peter | 0 | 1 | 0 | 1 | 1 | 0 |

# Règles d'association

- Une règle se note $X \Rightarrow Y$ ({Butter,Milk} $\Rightarrow$ {Milk})
- On utilise deux notions pour caractériser les fréquences des sous-ensembles (*itemset*)
  - Le support de $X \subseteq I$, défini comme la fraction des transactions dont $X$ est un sous-ensemble
  - La confiance, définie comme la probabilité conditionnelle que T contienne $Y$ sachant qu'elle contient $X$.
    Obtenue en divisant le support de $X \cup Y$ par celui de $X$
- On dit qu'une règle est une règle d'association de support minimal $s$ et de confiance minimale $c$ si :
  - Le support de $X \cup Y$ est d'au moins $s$
  - La confiance $X \Rightarrow Y$ est au moins $c$

# Règles d'association

Pour trouver les règles d'associations, on procède en 2 étapes :

- On cherche tous les *itemsets Z* de support $s > s_\theta$
- Pour chacun, on calcule toutes les partitions (X,Z-X), pour créer des règles $X \Rightarrow Z - X$
  On garde toutes les règles de confiance $c > c_\theta$
- $c_\theta$ et $s_\theta$ sont des seuils
- La première phase est coûteuse, c'est un champ de recherche (*frequent itemset mining*)

Adaptation au CF :

- Ces règles sont utiles avec des matrices unaires.
- On garde celles dont le 2nd membre contient 1 élément (exactement)
- On cherche les règles d'un utilisateur ($X \cup X_u$), on trie par confiance décroissante
- Les $k$ premiers éléments sont le top-$k$ pour cet utilisateur
- (plein de variantes)

# Classifieur Bayesien naïf

- On oublie l'ordre des notes, on considère qu'il s'agit de $l$ catégories $v_1 ... v_l$
- On cherche $r_{uj}$, avec une valeur parmi $v_1 ... v_l$
- On doit déterminer la probabilité que $r_{uj}$ prenne chacune de ses valeurs, sachant que l'on dispose d'un ensemble de notes $I_u$ :

$$P(r_{uj} = v_s | \text{notes de } I_u) \qquad \forall s \in \{v1, ..., v_l\}$$

- On transforme avec :

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(B)}$$

- en :

$$P(r_{uj} = v_s | \text{notes de } I_u) = \frac{P(r_{uj} = v_s) \cdot P(\text{notes de } I_u | r_{uj} = v_s)}{P(\text{notes de } I_u)}$$

- On gardera la plus grande probabilité

# Classifieur Bayesien naïf

- Le dénominateur ne dépend pas de $s$ :

$$P(r_{uj} = v_s | \text{notes de } I_u) \propto P(r_{uj} = v_s) \cdot P(\text{notes de } I_u | r_{uj} = v_s)$$

- $P(r_{uj} = v_s)$, *prior*, est la fraction des utilisateurs ayant donné $v_s$ comme note, parmi ceux ayant noté $j$
- Avec l'hypothèse *naïf* (indépendance des notes), on a :

$$P(\text{notes de } I_u | r_{uj} = v_s) = \Pi_{k \in I_u} P(r_{uk} | r_{uj} = v_s)$$

- chaque $P(r_{uk} | r_{uj} = v_s)$ est la fraction des utilisateurs qui ont noté $r_{uk}$ l'élément $k$, sachant qu'ils ont noté $v_s$ l'élément $j$

On peut ensuite estimer $\hat{r}_{uj}$ :

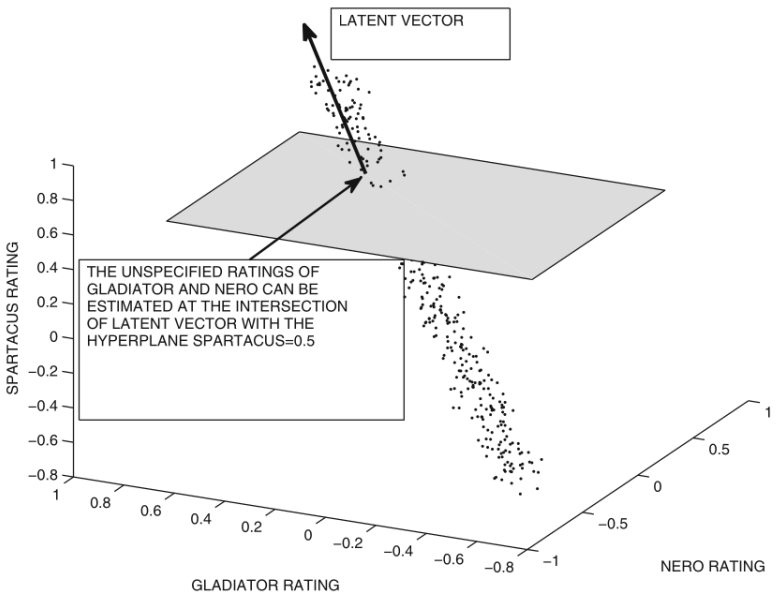
- en calculant les probabilités pour tous les $s$, prendre la plus grande, garder $v_s$
- raisonnable si $l$ est petit
- utiliser une moyenne pondérée, dont les poids des valeurs possibles sont les probabilités

# Plan

# Modèles à facteurs latents

- L'idée est de bénéficier du fait que des parties importantes des lignes/colonnes sont grandement corrélées (redondances)
- Pré-supposés :
  - On peut approximer la matrice complète avec une matrice de rang faible
  - Un sous-ensemble d'entrées de la matrice suffit pour obtenir une matrice complète de rang faible
  - Cette matrice de rang faible fournit des estimations robustes des entrées manquantes de la matrice initiale

# Intuition géométrique




- Notes corrélées, rang 1 (droite)
- Ici, une seule note (Spartacus 0.5) suffit pour estimer Nero et Gladiator !
- Intersection entre le vecteur latent (droite) et l'hyperplan (parallèle aux axes) tel que Spartacus ait 0.5

- En pratique : on n'a pas besoin de toute la matrice pour estimer les vecteurs latents principaux
- Orthogonalité des vecteurs latents

# Principe de la factorisation de matrices

- Une matrice $R$ de rang $k \ll \min(m, n)$ peut toujours être écrite sous la forme :

$$R = UV^T$$

- U est une matrice $m \times k$, V $n \times k$
- Les colonnes de U sont des vecteurs d'une base de l'espace de dim $k$ des colonnes de $R$
- Une ligne de V contient les coefficients pour combiner ces vecteurs en une colonne de $R$.
- Il existe un nombre infini de factorisations, correspondants à divers ensembles de vecteurs
- La SVD est un exemple, dans lequel il y a orthogonalité des colonnes/lignes

# Un peu d'algèbre linéaire

- si la matrice $R$ a un rang plus grand que $k$, on peut l'approximer par

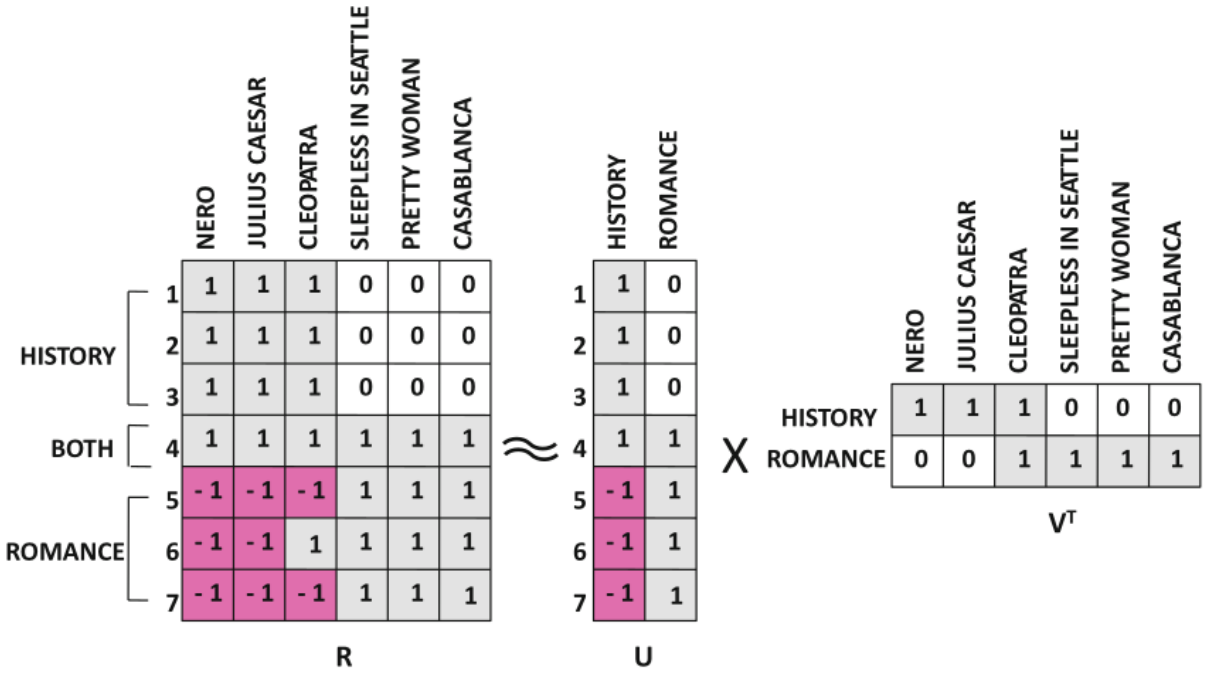
$$R \approx UV^T$$

- U est une matrice $m \times k$, V $n \times k$
- l'erreur d'approximation est

$$||R - UV^T||^2$$

- $||\cdot||^2$ représente la somme des carrés des entrées de $R - UV^T$, c'est la norme de Frobenius

# Un peu d'algèbre linéaire




- les utilisateurs 1-3 aiment les films historiques, neutre sur Romance
- 4 aime les 2 genres
- 4-7 aiment Romance, pas films historiques
- nombreuses corrélations !
- si on multiplie par -1, interprétations plus délicates

# Un peu d'algèbre linéaire

- chaque colonne de U est appelée vecteur latent, chaque ligne facteur latent
- la $i$e ligne $u_i$ de U est un facteur utilisateur, contenant "l'affinité" de l'utilisateur $i$ pour chacun des $k$ concepts (genres)
- chaque ligne $v_i$ de V est un facteur "item", contenant l'appartenance des films à chaque genre

# Apprentissage

- On cherche à résoudre :

$$\text{minimiser } J = \frac{1}{2}||R - UV^T||^2$$

- Sans contraintes sur U et V
- ”loss” quadratique, diverses SGD peuvent le résoudre

- MAIS : valeurs manquantes !
- Si on pouvait factoriser R en $UV^T$ complètes, on pourrait prédire les entrées manquantes de R :

$$\hat{r}_{ij} = \sum_{s=1}^{k} u_{is} \cdot v_{js}$$

- l’erreur pour un item est :

$$e_{ij} = rij - \hat{r}_{ij}$$

- d’où :

$$\text{minimiser } J = \frac{1}{2} \sum_{(i,j) \in S} e_{ij}^2$$

- noter le ”seulement sur les valeurs de S”

# Calculer : SGD

**Algorithm** *SGD*(Ratings Matrix: $R$, Learning Rate: $\alpha$)
**begin**
  Randomly initialize matrices $U$ and $V$;
  $S = \{(i,j) : r_{ij}$ is observed$\}$;
  **while** not(convergence) **do**
  **begin**
    Randomly shuffle observed entries in $S$;
    **for** each $(i,j) \in S$ in shuffled order **do**
    **begin**
      $e_{ij} \Leftarrow r_{ij} - \sum_{s=1}^{k} u_{is} v_{js}$;
      **for** each $q \in \{1 \ldots k\}$ **do** $u_{iq}^{+} \Leftarrow u_{iq} + \alpha \cdot e_{ij} \cdot v_{jq}$;
      **for** each $q \in \{1 \ldots k\}$ **do** $v_{jq}^{+} \Leftarrow v_{jq} + \alpha \cdot e_{ij} \cdot u_{iq}$;
      **for** each $q \in \{1 \ldots k\}$ **do** $u_{iq} = u_{iq}^{+}$ and $v_{jq} = v_{jq}^{+}$;
    **end**
    Check convergence condition;
  **end**
**end**

- U et V mises à jour simultanément, convergence plus rapide que batch
- SGD préférable pour données larges et temps de calcul goulet d'étranglement
- $\alpha = 0.005$ (learning rate), peut être adapté à chaque étape
- seuil de convergence à fixer astucieusement, trop d'itérations peuvent dégrader la qualité
- initialisation dans $(-1, 1)$ ?

# Régularisation

- Pénaliser les gros coefficients dans U et V
- Ajout d'un terme $\frac{\lambda}{2}(||U||^2 + ||V||^2), \lambda > 0$

# Principe de l'ALS

## Alternate Least Squares (Moindre Carrés Alternés)

- plus stable que SGD
- moins sensible à l'initialisation et la taille des pas
- moins efficace que SGD sur grandes données explicites
- version pondérée avec implicit feedback possible

## Principe

- U fixé, on traite le problème comme un problème de moindre carrés sur V. On détermine ainsi les $k$ facteurs latents de V. Il faut résoudre $n$ problèmes, mais ils sont indépendants (parallélisation)
- V fixé, on traite le problème comme un problème de moindre carrés sur U. On détermine ainsi les $k$ facteurs latents de U. Il faut résoudre $m$ problèmes, mais ils sont indépendants (parallélisation)

# NMF : non-negative Matrix Factorisation

- Outre la SVD, il existe d'autres factorisations, comme la NMF
- même fonction à optimiser, mais contraintes sur U et V :

$$U > 0, V > 0$$

- plus interprétable que d'autres méthodes
- marche bien avec échelles (1-5) mais surtout "like sans dislike" (implicit)

$$u_{ij} \Leftarrow \frac{(RV)_{ij} u_{ij}}{(UV^T V)_{ij} + \epsilon} \quad \forall i \in \{1 \dots m\}, \forall j \in \{1 \dots k\}$$
$$v_{ij} \Leftarrow \frac{(R^T U)_{ij} v_{ij}}{(VU^T U)_{ij} + \epsilon} \quad \forall i \in \{1 \dots n\}, \forall j \in \{1 \dots k\}$$

- $\varepsilon \sim 10^{-9}$
- les valeurs dasn U et V sont fixées à celles de la fin de l'itération précédente : mise à jour simultanées
- initialisation avec valeurs faibles dans $[0, 1]$.

# Factorisation de matrices : résumé

| Method | Constraints | Objective | Advantages/Disadvantages |
|---|---|---|---|
| Unconstrained | No constraints | Frobenius + regularizer | Highest quality solution<br>Good for most matrices<br>Regularization prevents overfitting<br>Poor interpretability |
| SVD | Orthogonal Basis | Frobenius + regularizer | Good visual interpretability<br>Out-of-sample recommendations<br>Good for dense matrices<br>Poor semantic interpretability<br>Suboptimal in sparse matrices |
| Max. Margin | No constraints | Hinge loss + margin regularizer | Highest quality solution<br>Resists overfitting<br>Similar to unconstrained<br>Poor interpretability<br>Good for discrete ratings |
| NMF | Non-negativity | Frobenius + regularizer | Good quality solution<br>High semantic interpretability<br>Loses interpretability with both like/dislike ratings<br>Less overfitting in some cases<br>Best for implicit feedback |
| PLSA | Non-negativity | Maximum Likelihood + regularizer | Good quality solution<br>High semantic interpretability<br>Probabilistic interpretation<br>Loses interpretability with both like/dislike ratings<br>Less overfitting in some cases<br>Best for implicit feedback |

# Recommandation par le contenu

# Recommandation Content-based

- Situations où l'on dispose de caractéristiques descriptives pour les éléments
- Exemple : genre de films, artistes d'une chanson (parolier, arrangeur, mélodiste, etc.), etc.
- Les seules notes de l'utilisateur suffisent !
- Particulièrement utile pour les éléments nouveaux (peu de notes)
- similarité avec ce que les utilisateurs ont déjà apprécié

Deux sources de données :

- les méta-données
- un profil utilisateur

# Étapes

1. Pré-traitement des éléments
   - Représentation vectorielle avec mots-clefs
   - Très dépendant du domaine
   - Poids, structure
   - Cf Recherche d'Information (NFE204)
   - Feature selection (Gini, $\chi^2$, entropie)
2. Modélisation des profils
   - Combinaison de feedback et d'attributs
   - Souvent proches des techniques classiques de classement ou régression
   - Pour chaque utilisateur, un modèle est entraîné (”documents étiquetés”)
   - Nearest-neighbors, Classifieur Bayesien naïf, Règles d'associations, ou régression (linéaire, par ex.)
3. Filtrage / recommandation

# Avantages, inconvénients

## Avantages

- Les autres ne comptent plus !
- Cold start pour les éléments
- Explications grâce aux caractéristiques des éléments
- Les classifieurs textuels abondent, facilité d'implémentation et d'intégration dans des systèmes existants

## Inconvénients

- Sur-spécialisation : enfermement dans des "bulles", absence de nouveauté
- Cold-start utilisateur
- Pour les utilisateurs, il faut un certain nombre de notes pour éviter le sur-apprentissage

# Systèmes de recommandation hybrides

# Recommandation hybride

- Les systèmes de filtrage par le contenu et de filtrage collaboratif ont des performances avantages et inconvénients divers
- Des sources de données diverses

- Combiner plusieurs systèmes est presque une évidence
- On cherche à compenser les faiblesses d'un système par un autre
- On parle de systèmes hybrides

# Recommandation hybride

Il y a plusieurs façons de construire des systèmes hybrides :

- L'approche "ensembliste", où on combine les résultats de plusieurs systèmes sans les modifier (un content-based et un CF, par exemple). Utilisée traditionnellement en ML.
- L'approche monolothique, dans laquelle on tente de modifier des algorithmes existants avec des caractéristiques d'autres systèmes (un collaborative filtering avec prise en compte d'attributs)

On peut combiner des systèmes de même type (exemple : plusieurs content-based). Les systèmes ayant gagné le Netflix prize combinaient des approches de voisinage et de facteurs latents.

# Recommandation hybride

Il y a également différentes manières de combiner :

- approche ”weighted” : pondération des scores données par plusieurs systèmes
- approche ”switching” : selon les possibilités et les besoins, on utilise l'un ou l'autre des systèmes de reco mobilisé (exemple : éviter le cold-start, ou choisir le système avec la meilleure accuracy)
- approche cascade : un système de reco raffine les recommandations proposées par un autre (biais dans l'apprentissage), avant une sortie unifiée
- approche ”feature augmentation” : les résultats d'un système servent d'entrée (supplémentaire) pour le suivant

- approche meta : le modèle d'un système sert d'entrée à un autre. Exemple : content-based pour améliorer la sélection de voisinage en CF.

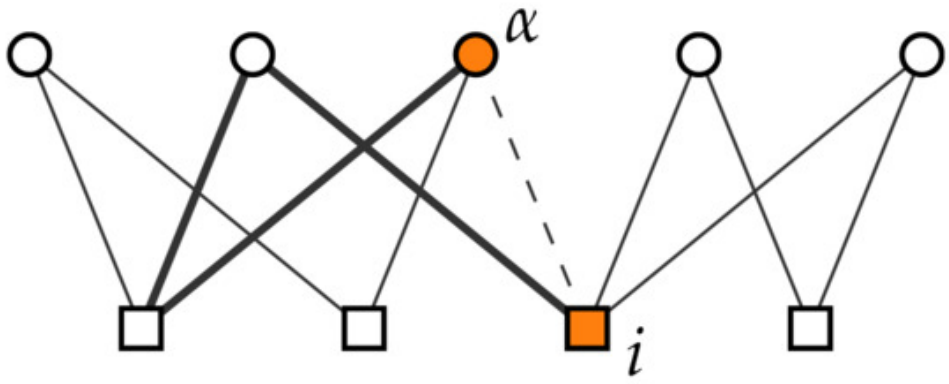
# Recommandation sociale

# Recommandation sociale

”Social” a de nombreux sens dans le domaine de la recommandation, différents et importants :

- Le contexte social de l’utilisateur :
  - Temps : les recommandations devraient être différentes selon le contexte temporel (heure de la journée, semaine de l’année, jour de la semaine, etc.)
  - Lieu : les recommandations gagnent en précision en tenant compte du lieu (voyage, achats, etc.)
  - Entourage : cinéma (film différent en famille, avec des amis)
- Le ”réseau social” :
  - Autorité, importance (PageRank)
  - Prédiction de liens dans un graphe (biparti)
  - Influence
  - Réseau de confiance

# Prédiction de liens




- Internet, le Web, et les réseaux sociaux ont donné une structure forte à de nombreuses activités humaines
- faire de la recommandation peut être vu comme ”chercher à prévoir les liens pertinents pour le réseau... mais pas encore apparus
- dans certains cas, comme Facebook, susciter des liens est même central pour la plateforme

# Prédiction de liens

Plusieurs indicateurs peuvent être calculés pour estimer la probabilité qu'un lien apparaisse :

- *Common neighbors* : nombre de voisins en communs

$$CN(i,j) = |N_i \cap N_j|$$

non normalisé (problème des grands degrés)

- Indice Jaccard (normalisé pour les degrés de i et j) :

$$\text{Jacc}(i,j) = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$$

mais dans les voisinages, les degrés peuvent être forts

- Adamic-Adar (version pondérée de CN) :

$$\text{AdamicAdar}(i,j) = \sum_{k \in N_i \cap N_j} \frac{1}{log(N_k)}$$

(base log peu importante)

# Prédiction de liens

Quand il y a peu de liens, on peut s'en remettre à des mesures reposant sur les chemins dans le graphe :

- Mesure de centralité (Katz)
- Mesure avec marches aléatoire (personalized PageRank)

On peut aussi approcher la prédiction de liens comme un problème de classification (supervisée) :

- Création d'un dataset avec de features (similarités diverses, degrés, etc.)
- La présence/absence d'un lien est une variable binaire
- Une régression logistique, modifiée pour tenir compte du déséquilibre de classes, peut être utilisée

Enfin, on peut approcher le problème avec de la factorisation de matrices.

# Influence

- Dans un réseau d'interactions humaines, le bouche à oreille permet la propagation de messages
- Certains messages se diffusent plus (vite) que d'autres (cascades)
- Analogie avec l'épidémiologie, les réseaux informatiques (diffusion)
- Les facteurs les plus importants pour la diffusion :
  - La centralité de l'acteur
  - La force des liens (amitiés longues, statut social)

- Problème de recherche : *influence propagation models*.
  On cherche un ensemble de nœuds S (de taille k) tel que la propagation de l'information soit maximisée dans le réseau (par influence)
  - Modèle à seuil
  - Modèle à propagation unique

# Confiance, homophilie

- Utiliser la structure du réseau social ”amical” des utilisateurs
- Homophilie : les utilisateurs sont généralement connectés à d’autres utilisateurs ayant les mêmes goûts
- Les recommandations de proches sont plus fiables, a priori, que celles de n’importe qui
- Ce ”réseau social” n’est pas toujours existant, mais peut aider face au cold-start.
- On peut essayer d’inférer la confiance entre utilisateurs, à partir des liens existants. Il existe des plateformes où l’on demande explicitement aux utilisateurs de dire à quels utilisateurs ils font confiance (Epinions).

La confiance peut permettre de trouver/modifier les peer-groups à partir desquels on calcule les moyennes pour prédire les notes.

# Bilan de la recommandation sociale

## Avantages

- Amélioration de la qualité des recommandations proposées
- Notamment : les utilisateurs ”controverse”, ceux qui ne sont pas d’accord avec les autres
- Cold start : si on dispose du réseau social mais pas de notes, on peut proposer des recommandations aux nouveaux utilisateurs
- Résistance aux attaques : la recommandation repose sur des liens ”fiables”, il est plus dur de piéger le système avec des ”faux profils” (Amazon), puisque les faux profils ne sont en général pas liés aux utilisateurs réels.

## Inconvénients

- Difficulté d’avoir des données

# Évaluation des systèmes de recommandation

# Évaluation de systèmes de recommandation

- Méthodes ”online” (à chaud)
  - A/B testing (taux de conversion)
  - Complexité dans les grands systèmes
- Méthodes ”offline” (à froid)
  - Datasets larges, connus, robustes
- Études utilisateurs (user studies)

## Remarques

- Similitude avec l’évaluation de la classification/régression
- Toutefois, aspect ”pratique” : multi-facettes
- Il faut un processus clair pour avoir la ”bonne” évaluation

# Études utilisateurs

- Des sujets d'études sont recrutés, doivent accomplir des tâches précises
- Retours collectés avant et après, et résultat de l'interaction
- Notes, classement, appréciations générales

## Avantages

- Divers scénarios / interfaces testables
- Précision du retour

## Inconvénients

- Utilisateurs averti que "c'est un test" (biais)
- Difficulté à recruter /tester de nombreux utilisateurs (représentativité de l'échantillon)

# Évaluation à chaud

- Avec "les *vrais* utilisateurs" du système en place
  - Moins de biais (échantillon)
  - Souvent employé pour comparer divers algorithmes
- A/B testing :
  - On prend plusieurs (deux) échantillons de même tailles/profils
  - Chaque algo est testé sur un échantillon
  - Un algo est souvent "une modification de l'ancien" et l'autre algo est "celui en place"

## Remarques

- On compare souvent avec un "taux de conversion" (clic sur des articles, par exemple)
- Similitude avec les essais cliniques en médecine (essai randomisé contrôlé, ou randomized controlled trial ou RCT), avec "groupe contrôle"
- La segmentation peut ne pas être stricte (un utilisateur reçoit A ou B aléatoirement)

# Évaluation à chaud

## Avantages

- Qualité des observations
- Lien avec apprentissage par renforcement et multi-armed bandits (optimisation gloutonne)

## Inconvénients

- Il faut en général beaucoup d'utilisateurs dans le système pour que ça marche
- Il faut la main sur le système pour mettre en place les tests
  - Évaluation académique (reproductibilité) limitée
  - Robustesse / généralisabilité réduites (manque de tests)

# Évaluation à froid

- On utilise les données compilées sur un système
- Notes, user/item id, horodatage éventuel (peu de détails)
- En général, ça peut devenir un standard (réutilisation académique/industrie) :
  - Netflix Prize
  - MovieLens
  - LastFm, Flixster, etc.

# Objectifs de l'évaluation

- Accuracy (justesse, pertinence)
  - métriques de régression, classification : MSE/RMSE
- Coverage (couverture)
  - sparsity
  - peut-être artificiellement réduite (prédiction par défaut)
  - % des utilisateurs/items pour lesquels on peut prédire $k$ éléments
- Nouveauté
  - recommandations non vues avant / non connues
- "Sérendipité" (nouveau ET "moins évident")
- Diversité
  - Listes "un peu" variées (sinon, risque d'échec)
  - Améliore nouveauté et sérendipité
- Robustesse (résister aux attaques)
- Scalabilité (temps de calcul)

# Justesse

- MSE

$$MSE = \frac{\sum_{(u,j)\in E}(\hat{r}_{uj} - r_{uj})^2}{|E|}$$

- Plus c'est bas, moins il y a d'erreur, mieux c'est
- RMSE : ramené en terme de notes.
- (R)MSE : pénalisé par grosses erreurs !
- MAE : Mean Absolute Error

$$MAE = \frac{\sum_{(u,j)\in E}|\hat{r}_{uj} - r_{uj}|}{|E|}$$

- Version normalisée entre 0 et 1 (en divisant par $r_{max} - r_{min}$)

# Évaluation des classements

- Corrélation de classements
  - Pearson
  - Kendall (+1/-1 si même ordre, sommés pour toutes paires d'items pour $u$)
- Utilité : corriger l'impact du "bas du classement"
  - (N)DCG : Discounted Cumulative Gain :

$$DCG = \frac{1}{m} \sum_{u=1}^{m} \sum_{j \in I_u} \frac{2^{rel_{uj}} - 1}{\log_2(v_j + 1)}$$

$rel_{uj}$ : pertinence (note brute)

  - MAP@k (Mean Average Precision) : fraction d'éléments pertinents pour une liste de k éléments pour chaque utilisateur, moyennée
- AUC : Area Under (Receiver Operating) Curve
  - évaluations binaires

# Thématiques avancées

# LLM

Les LLM (*large language models*) changent la recommandation, à divers niveaux :

- Utilisation avancée de la langue :
  - Recommandation reposant sur du texte : ”un film comme Inception, mais plus léger”
  - Recommandation avec peu d’infos mais un prompt élaboré
- Données non structurées :
  - avis sur des produits
  - commentaires et tweets
- Modalités :
  - chatbot et conversation
  - adaptation temps réel

## Nouveau paradigme ?

- un peu : *sequence-based* recommandation, l’historique est une phrase et on prédit le prochain ”mot”
- mais surtout : inclusion à divers niveaux dans des systèmes existants

# Sujets avancés non traités

- Prise en compte élaborée du temps et du lieu
- Recommandation de groupe
- Résistance aux attaques
- Gestion élaborée de la vie privée

# Références

# Merci

Ce cours repose notamment sur :

- le livre Recommender systems, the textbook, de Charu C. Aggarwal. Springer, 2016.